UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/986,019	11/07/2001	Stefan Dyckerhoff	0023-0051	4743

44987 7590 01/25/2005

HARRITY & SNYDER, LLP
11240 WAPLES MILL ROAD
SUITE 300
FAIRFAX, VA 22030

EXAMINER	
GERSTL, SHANE F	

ART UNIT	PAPER NUMBER
2183	

DATE MAILED: 01/25/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application N .	Applicant(s)	
	09/986,019	DYCKERHOFF ET AL.	
	Examiner	Art Unit	
	Shane F Gerstl	2183	

-- The MAILING DATE of this communication appears n the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *29 October 2004*.

2a)☒ This action is **FINAL**. 2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-4 and 6-31* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-4 and 6-31* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *07 November 2001* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. Claims 1-31 have been examined.

Papers Received

2. Receipt is acknowledged of amendment papers submitted, where the papers have been placed of record in the file.

3. The objections to the drawings and claims as well as the 36 USC 112 rejections have been overcome by the amendment and are herein withdrawn.

Claim Rejections - 35 USC § 102

4. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

5. Claims 21, 22, and 24-26 are rejected under 35 U.S.C. 102(e) as being anticipated by Nemirovsky (6,477,562).

6. In regard to claim 21, Nemirovsky discloses a method for processing a packet (column 5, lines 10-21 and column 12, lines 27-31) to determine control information for the packet, the method comprising:

a. reading a plurality of instructions; [Note column 6, line 65 – column 7, line 11. This section shows that instructions are transferred from the instruction cache to a multi-threaded fetch unit (figure 2, elements 202 and 203) where they

are stored in one or more queues and thus, the instructions are read by this fetch unit.]

b. generating a predicted address based on a predetermined one of the read instructions; [Column 6, lines 9-16 show that branch prediction is used and thus a predicted address is generated and is inherently based on a branch instruction.]

c. evaluating the read instructions; [Column 6, line 65 – column 7, line 11 show that in one embodiment, each queue is partitioned into units for each stream and thus the individual instructions are inherently evaluated to see what stream they belong to.]

d. selecting one of the read instructions based on the evaluations; [Figure 2 shows that the instructions are then passed from the fetch unit into an instruction scheduler. Column 7, lines 39 – 67 show that streams (and inherently the instructions therein) are then selected. Thus the instructions (including the first (a single) instruction) are selected based on their associated stream, which was evaluated in the instruction fetch mechanism by placement into the corresponding queue. In an example where four instructions belonging to four different streams are fetched, only one will be selected at a time based on the evaluation of it and what stream it belongs to.]

e. and performing operations related to determining the control information for the packet based on the selected instruction, the operations including generating a next address for reading instructions. [As given in column 6, lines 9-16, Nemirovsky shows that in some embodiments priority determination is the

result of branch prediction. Column 1, lines 56-60 further show that branch prediction units are used in the art of the invention. In order to have a branch prediction, a branch instruction must inherently be fetched or read as given above. The included dictionary definition of "branch prediction" shows that it is guessed or predicted whether a branch will be taken or not and code is prefetched from the appropriate location (address). Thus the destination address is predicted (generated) and instructions are executed from that location based on having read and evaluated a branch instruction.]

7. In regard to claim 22, Nemirovsky discloses the method of claim 21, wherein evaluating the read instructions is performed based on a field in the instructions that specifies a logical operation that is to be performed. [The enclosed definition of "header" shows that it is a portion of the packet that contains source and destination addresses and other fields. The included IEEE definition shows that a header is all of this plus also gives the type of frame or packet. Column 12, lines 19-34 show that priority is set or evaluated based on the type of thread or packet being handled. Since the circuitry to do so is inherently made up of logic gates, a logic function is performed.]

8. In regard to claim 24, Nemirovsky discloses the method of claim 21, wherein the selecting of one of the read instructions is performed as a priority selection based on a read instruction that evaluates to a logic true value. [Column 7, lines 57-60 show that since there are multiple streams, there is a mechanism for selecting the stream and this is based on priority. Column 5, lines 10-21 show that the priority (evaluation results) is dynamically determined or evaluated and is based on each stream and thus the

instructions therein. The instruction that is being selected is being interpreted as the logic true value since it has the highest evaluated priority.]

9. In regard to claim 25, Nemirovsky discloses the method of claim 21, wherein the operations include an extract instruction that is used to extract designated information from the packet into a memory. [Figure 2 illustrates a load/store unit and thus load instructions are executed, which inherently extract information from memory based on information extracted from the load instruction of the packet and then stored in a register or other local memory.]

10. In regard to claim 26, Nemirovsky discloses the method of claim 21, wherein the operation includes a write instruction used to write information contained in a field of the write instruction to a memory. [Figure 2 illustrates a load/store unit and thus store instructions are executed, which inherently write information from a field indicated by the instruction into memory.]

Claim Rejections - 35 USC § 103

11. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

12. Claims 1-4, 6-9, and 23 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nemirovsky in view of Hennessy (Computer Organization and Design).

13. In regard to claim 1,

a. Nemirovsky discloses a pipelined processor comprising:

i. An instruction memory (figure 2, element 202);

ii. a program counter configured to provide the address for reading instructions from the plurality of instruction memories; [Column 7, lines 3-7 show that instructions are fetched from the instruction cache (memory). Column 7, lines 14-19 show that a program counter exists for multiple streams or threads. The included dictionary definition of "program counter" shows that a program counter points to the next instruction to be fetched or read.]

iii. a priority encoder configured to select one of the instructions based on evaluation results generated from the instructions read from the instruction memories; [Column 7, lines 57-60 show that since there are multiple streams, there is a mechanism for selecting the stream and this is based on priority. Column 5, lines 10-21 show that the priority (evaluation results) is dynamically determined or evaluated and is based on each stream and thus the instructions therein. The included definitions of "encoder" and "encode" show that an encoder is simply a circuit that converts data into a given format and the selecting mechanism above does so by converting multiple inputs into a single selected output.]

iv. and an execution unit configured to receive the selected one of the instructions and to perform operations indicated by the selected instruction (figure 2, elements 207-210).

b. Nemirovsky does not explicitly disclose

v. a plurality of memories;

vi. pipelining in general nor a first pipeline stage including the memories and program counter and a second pipeline stage including the priority encoder and execution unit.

c. Hennessy has taught in figure 6.12 on page 452 a first pipeline stage (IF) including an instruction memory and program counter (PC) and a second pipeline stage (EX) that contains an execution unit (ALU) and logic to select appropriate instruction data (MUX). Hennessy has also taught on pages 576-578 the use of a second level cache. One of ordinary skill in the art would have recognized that a second level cache is addressed using portions or tags of the same address (program counter address for an instruction cache) as the first level cache uses and thus a program counter corresponds to a location in each of the instruction caches or instruction memories.

d. Hennessy teaches on page 436 that pipelining speeds up execution. This execution speed-up taught by Hennessy would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to use the pipeline disclosed in Hennessy. Hennessy also teaches on page 576 that the use of a secondary cache gives improved performance. This improved performance would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to use a second level cache as taught by Hennessy. With such a pipeline in Nemirovsky, the instruction cache (element 202 of figure 2) and the fetch unit

(within element 203) would be in a first pipeline stage while the execution units (elements 207-210), issue network (element 6), and instruction scheduler (element 5) would be in a second pipeline stage.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Nemirovsky to use the pipeline of Hennessy so the processor is sped-up as taught by Hennessy and to use a second level instruction cache for further performance gains as also taught by Hennessy.

14. In regard to claim 2, Nemirovsky in view of Hennessy discloses the pipelined processor of claim 1, wherein the execution unit performs packet processing operations. [As shown in column 5, lines 10-21, the processor is used for packet processing and thus executes packet processing operations.]

15. In regard to claim 3, Nemirovsky in view of Hennessy discloses the pipelined processor of claim 1, further comprising: a packet header buffer connected to the priority encoder and the execution unit. [The enclosed definition of "header" shows that it is a portion of the packet that contains source and destination addresses and other fields. Therefore, the header, being a part of the packet, must be stored like the packet in the memories (instruction cache), pipeline buffers (figure 6.12 of Hennessy), and prefetch buffers of figure 2.]

16. In regard to claim 4, Nemirovsky in view of Hennessy discloses the pipelined processor of claim 1, wherein the instructions are read from a memory address equal to the value of the program counter (as shown above).

17. In regard to claim 6, Nemirovsky in view of Hennessy discloses the pipelined processor of claim 1, wherein the evaluation components generate the evaluation results based on a logical operation dictated by the instructions. [Column 6, lines 50-60 show that the priority control unit changes or evaluates the priority and is comprised of logic and thus performs a logical operation. As shown above, the priority is associated with the instructions in the streams and in a specific embodiment of column 12, lines 27-31, for example, it is shown that the type of thread or stream (and thus the type of instructions) control or dictate the priority.]

18. In regard to claim 7, Nemirovsky in view of Hennessy discloses the pipelined processor of claim 1 further comprising:

a. a branch prediction component (column 6, lines 9-16) configured to generate a predicted program counter value based on instructions read from one of the instruction memories; [Since branch prediction gives a prediction of a branch instruction, the prediction is inherently based on a branch instruction, which must be read from an instruction memory. Also, since the system of Nemirovsky in view of Hennessy points to instructions to fetch with a program counter and branch prediction inherently produces a predicted next address the (whether the branch is taken or not taken), the prediction must produce a predicted program counter.]

b. wherein the execution unit generates a true program counter value based on the selected instruction and generates an indication of whether the predicted program counter value is accurate. [Branch prediction inherently must be

checked with the outcome of the execution of the branch so it is known whether or not the prediction was correct or accurate so that the correct code is certain to be executed. Since the system has a branch execution unit (figure 2, element 207), this is where the branch execution takes place that reveals the true next instruction address or program counter address.]

19. In regard to claim 8, Nemirovsky in view of Hennessy the pipelined processor of claim 1, wherein the second pipeline stage further comprises: a multiplexer configured to receive the read instructions and to forward the selected one of the instructions to the execution unit based on a signal from the priority encoder. [As described above, the selection of the instructions in the second pipeline stage is based on a priority and the selection unit chooses an instruction at one input from many and outputs it (the definition of a multiplexer) for execution.]

20. In regard to claim 9, Nemirovsky in view of Hennessy the pipelined processor of claim 1, further comprising: a plurality of memory elements implemented as an interface between the first and second pipeline stages. [The pipeline of Hennessy in figure 6.12 shows that there are memory elements between each pipeline stage.]

21. In regard to claim 23,

a. Nemirovsky discloses the method of claim 21,

b. Nemirovsky does not explicitly disclose wherein the method is performed in two pipelined stages.

c. Hennessy has taught in figure 6.12 on page 452 a pipelined processor with at least two pipelined stages.

d. Hennessy teaches on page 436 that pipelining speeds up execution. This execution speed-up taught by Hennessy would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to use the pipeline disclosed in Hennessy.

22. It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Nemirovsky to use the pipeline of Hennessy so the processor is sped-up as taught by Hennessy.

23. Claims 10-20, and 28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nemirovsky in view of Hennessy (Computer Architecture) and Hennessy (Computer Organization and Design).

24. In regard to claim 10,

a. Nemirovsky discloses a network device comprising:

i. a physical interface configured to receive packets from and transmit packets to a network; [Column 12, lines 20-24 show that flows of packets are received from and transmitted to a network.]

ii. and a processing unit configured to store the received packets (Figure 2) and to examine header information of the packets, [The enclosed definition of "header" shows that it is a portion of the packet that contains source and destination addresses and other fields. The included IEEE definition shows that a header is all of this plus also gives the type of frame or packet. Column 12, lines 19-34 show that priority is set based on

the type of thread or packet being handled. Since this type is stored in the header of the packet, the header must be examined.]

iii. the processing unit including a processing engine that comprises:

(1) reading a plurality of packet processing instructions relating to processing of a first packet from instruction memories, [Column 7, lines 3-7 show that the instructions are fetched from the instruction cache (memory). Since the processor has been shown in column 5, lines 10-21 to process packets, it inherently reads and executes packet processing instructions, which relate to at least a first packet. As shown in figure 2, instructions are received from multiple memories (cache memory or main memory on a cache miss) each cycle. Note column 6, line 65 – column 7, line 11. This section shows that instructions are transferred from the instruction cache to a multi-threaded fetch unit (figure 2, elements 202 and 203) where they are stored in one or more queues and thus, the instructions are read by this fetch unit. This section further shows that in one embodiment, each queue is partitioned into units for each stream and thus the individual instructions are inherently evaluated to see what stream they belong to.]

(2) selecting one of the packet processing instructions for execution; [Column 7, lines 57-60 show that since there are multiple streams, there is a mechanism for selecting the stream,

which is then executed by an execution unit of figure 2 (elements 207-210). Column 7, lines 39 – 67 show that streams (and inherently the instructions therein) are then selected. Thus the instructions (including the first (a single) instruction) are selected based on their associated stream, which was evaluated in the instruction fetch mechanism by placement into the corresponding queue. In the example where four instructions belonging to four different streams are fetched, only one will be selected at a time based on the evaluation of it and what stream it belongs to.]

b. Nemirovsky does not explicitly disclose the processing engine where the plurality of instructions are read per cycle or the engine being a pipelined packet processing engine that comprises:

iv. a first pipeline stage containing the read function;

v. and a second pipeline stage containing the select function.

c. Hennessy (COD) has taught in figure 6.12 on page 452 an instruction fetch (read) pipeline stage (IF) including an instruction memory and a second pipeline stage (EX) that contains an execution unit (ALU) and logic to select appropriate instruction data (MUX). Hennessy (CA) has shown on pages 278-284 and 335-349 the basics of and a specific example of a superscalar processor where a processor fetches (reads) multiple instructions simultaneously in the same cycle (page 280, 341, and 342), which is need for issuing multiple instructions per cycle.

d. Hennessy (COD) teaches on page 436 that pipelining speeds up execution. This execution speed-up taught by Hennessy would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to use the pipeline disclosed in Hennessy. With such a pipeline in Nemirovsky, the instruction cache (element 202 of figure 2) and the fetch unit (within element 203) would be in a first pipeline stage while the execution units (elements 207-210), issue network (element 6), and instruction scheduler (element 5) would be in a second pipeline stage. Hennessy (CA) has shown on page 278 that issuing multiple instructions per cycle (and thus fetching or reading multiple instructions simultaneously) provides performance improvements. This improved performance would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to fetch and issue multiple instructions simultaneously as taught by Hennessy (CA).

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Nemirovsky to use the pipeline of Hennessy (COD) so the processor is sped-up as taught by Hennessy (COD) and to fetch and issue multiple instructions simultaneously as taught by Hennessy (CA) so that performance is improved as also taught by Hennessy (CA).

25. In regard to claim 11, Nemirovsky in view of Hennesy and Hennessy discloses the network device of claim 10, wherein the network device is a router (column 12, lines 19-31 and column 5, lines 10-21).

26. In regard to claim 12, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 10, wherein the first pipeline stage comprises: a program counter configured to store a program address value used to read the packet processing instructions from the instruction memories. [Column 7, lines 3-7 show that instructions are fetched from the instruction cache (memory). Column 7, lines 14-19 show that a program counter exists for multiple streams or threads. The included dictionary definition of "program counter" shows that a program counter points to the next instruction to be fetched or read. The pipeline of Hennessy shows that the program counter (PC) is in the first stage.]

27. In regard to claim 13, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 10, wherein the second pipeline stage comprises: a priority encoder configured to select the one of the packet processing instructions based on evaluation results generated from the packet processing instructions read from the instruction memories, and an execution unit (figure 2, elements 207-210) configured to receive the selected one of the packet processing instructions and to perform operations indicated by the selected packet processing instruction. [Column 7, lines 57-60 show that since there are multiple streams, there is a mechanism for selecting the stream and this is based on priority. Column 5, lines 10-21 show that the priority (evaluation results) is dynamically determined or evaluated and is based on each stream and thus the instructions therein. The included definitions of "encoder" and "encode" show that an encoder is simply a circuit that converts data into a given format

and the selecting mechanism above does so by converting multiple inputs into a single selected output.]

28. In regard to claim 14, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 12, wherein the plurality of instructions read from the instruction memories are read from a memory address equal to the value of the program counter (as shown above).

29. In regard to claim 15, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 13, wherein the second pipeline stage further comprises: an evaluation component corresponding to each of the instruction memories, the evaluation components generating the evaluation results based on each of the instructions read from the instruction memories. [Column 5, lines 10-21 show that the priority codes (evaluation results as above) are associated with the streams and dynamically determined by a priority controller or evaluation component. Since there is only one priority controller in the embodiment of figure 2, element 9, it must be associated with each of the plurality of instruction memories.]

30. In regard to claim 16, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 15, further comprising: a packet header buffer connected to the evaluation component. [The enclosed definition of "header" shows that it is a portion of the packet that contains source and destination addresses and other fields. Therefore, the header, being a part of the packet, must be stored like the packet in the memories (instruction cache), pipeline buffers (figure 6.12 of Hennessy), and prefetch buffers of figure 2.]

31. In regard to claim 17, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 15, wherein the evaluation components generate the evaluation results based on a logical operation dictated by the packet processing instructions. [Column 6, lines 50-60 show that the priority control unit changes or evaluates the priority and is comprised of logic and thus performs a logical operation. As shown above, the priority is associated with the instructions in the streams and in a specific embodiment of column 12, lines 27-31, for example, it is shown that the type of thread or stream (and thus the type of instructions) control or dictate the priority.]

32. In regard to claim 18, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 10, further comprising:

a. a branch prediction component (column 6, lines 9-16) configured to generate a predicted program counter value based on packet processing instructions read from one of the instruction memories, [Since branch prediction gives a prediction of a branch instruction, the prediction is inherently based on a branch instruction, which must be read from an instruction memory. Also, since the system of Nemirovsky in view of Hennessy points to instructions to fetch with a program counter and branch prediction inherently produces a predicted next address the (whether the branch is taken or not taken), the prediction must produce a predicted program counter.]

b. wherein an execution unit generates a true program counter value based on the selected packet processing instruction and generates an indication of whether the predicted program counter value is accurate. [Branch prediction

inherently must be checked with the outcome of the execution of the branch so it is known whether or not the prediction was correct or accurate so that the correct code is certain to be executed. Since the system has a branch execution unit (figure 2, element 207), this is where the branch execution takes place that reveals the true next instruction address or program counter address.]

33. In regard to claim 19, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 13, wherein the second pipeline stage further comprises: a multiplexer configured to receive the read packet processing instructions and to forward the selected one of the packet processing instructions to an execution unit based on a signal from the priority encoder. [As described above, the selection of the instructions in the second pipeline stage is based on a priority and the selection unit chooses an instruction at one input from many and outputs it (the definition of a multiplexer) for execution.]

34. In regard to claim 20, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 10, further comprising: a plurality of timing buffers implemented as an interface between the first and second pipeline stages. [The pipeline of Hennessy in figure 6.12 shows that there are memory elements between each pipeline stage.]

35. In regard to claim 31, Nemirovsky in view of Hennessy and Hennessy discloses the network device of claim 10, wherein the non-selected one of packet processing instructions are not executed. [As shown previously, the selection picks a single stream

of instructions to execute, thus the unselected streams and instructions will not be executed at least until a new selection takes place in the next cycle.]

36. In regard to claim 28, Nemirovsky discloses

a. The pipelined processing device of claim 27,

b. Nemirovsky in view of Hennessy (CA) does not disclose wherein the means for reading and the means for selecting are implemented as first and second stages of the pipeline, respectively.

c. Hennessy (COD) has taught in figure 6.12 on page 452 an instruction fetch (read) pipeline stage (IF) including an instruction memory and a second pipeline stage (EX) that contains an execution unit (ALU) and logic to select appropriate instruction data (MUX).

d. Hennessy (COD) has taught on page 436 that pipelining speeds up execution. This execution speed-up taught by Hennessy would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to use the pipeline disclosed in Hennessy. With such a pipeline in Nemirovsky, the instruction cache (element 202 of figure 2) and the fetch unit (within element 203) would be in a first pipeline stage while the execution units (elements 207-210), issue network (element 6), and instruction scheduler (element 5) would be in a second pipeline stage.

It would have been obvious to one of ordinary skill in the art at the time of invention to modify the design of Nemirovsky in view of Hennessy (CA) to use the pipeline of Hennessy (COD) so the processor is sped-up as taught by Hennessy (COD).

37. Claims 27, 29, and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable over Nemirovsky in view of Hennessy (Computer Architecture).

38. In regard to claim 27,

a. Nemirovsky has disclosed a pipelined processing device (figure 2) comprising:

i. means for reading a plurality of processing instructions from instruction memory that relate to processing a packet; [Column 7, lines 3-7 show that instructions are fetched or read from the instruction cache (memory). Column 5, lines 10-21 shows an embodiment where the device processes packet data.]

ii. and means for selecting one of the read instructions for execution based on a priority encoding of evaluation results related to each of the read instructions. [Column 7, lines 39 – 67 show that streams (and inherently the instructions therein) are then selected. Thus the instructions (including the first (a single) instruction) are selected based on their associated stream, which was evaluated in the instruction fetch mechanism by placement into the corresponding queue. In an example where four instructions belonging to four different streams are fetched, only one will be selected at a time based on the evaluation of it and what stream it belongs to.]

b. Nemirovsky does not explicitly disclose simultaneously reading the instructions from memory.

c. Hennessy has shown on pages 278-284 and 335-349 the basics of and a specific example of a superscalar processor where a processor fetches (reads) multiple instructions simultaneously in the same cycle (page 280, 341, and 342), which is need for issuing multiple instructions per cycle.

d. Hennessy has shown on page 278 that issuing multiple instructions per cycle (and thus fetching or reading multiple instructions simultaneously) provides performance improvements. This improved performance would have motivated one of ordinary skill in the art to modify the design of Nemirovsky to fetch and issue multiple instructions simultaneously as taught by Hennessy.

It would have been obvious to one of ordinary skill in the art to modify the design of Nemirovsky to fetch and issue multiple instructions simultaneously as taught by Hennessy so that performance is improved as also taught by Hennessy.

39. In regard to claim 29, Nemirovsky in view of Hennessy discloses the pipelined processing device of claim 27, further comprising: means for storing a value that designates an address to the instruction memory. [Column 7, lines 3-7 show that instructions are fetched from the instruction cache (memory). Column 7, lines 14-19 show that a program counter exists for multiple streams or threads. The included dictionary definition of "program counter" shows that a program counter points to the next instruction to be fetched or read.]

40. In regard to claim 30, Nemirovsky in view of Hennessy discloses the pipelined processing device of claim 29, further comprising: means for generating a predicted next value to store in the means for storing; and means for generating a true next value

to store in the means for storing. [As shown above, Nemirovsky discloses the use of branch prediction and the target address of a predicted branch is inherently generated (in some fashion) and stored in the program counter. For sequential instructions, the program counter will inherently receive and store a generated (incremented or otherwise) true next value (non-predicted value).]

Response to Arguments

41. Applicant's arguments filed 10/29/04 have been fully considered but they are not persuasive.

42. Applicant has argued with respect to claims 1, 13, and 21 that Nemirovsky does not disclose or suggest evaluating a plurality of read instructions and selecting one of the read instructions based on the evaluations. Note column 6, line 65 – column 7, line 11. This section shows that instructions are transferred from the instruction cache to a multi-threaded fetch unit (figure 2, elements 202 and 203) where they are stored in one or more queues and thus, the instructions are read by this fetch unit. This section further shows that in one embodiment, each queue is partitioned into units for each stream and thus the individual instructions are inherently evaluated to see what stream they belong to. Figure 2 shows that the instructions are then passed from the fetch unit into an instruction scheduler. Column 7, lines 39 – 67 show that streams (and inherently the instructions therein) are then selected. Thus the instructions (including the first (a single) instruction) are selected based on their associated stream, which was evaluated in the instruction fetch mechanism by placement into the corresponding queue. In an example where four instructions belonging to four different streams are

fetched, only one will be selected at a time based on the evaluation of it and what stream it belongs to. The above interpretation is how the Examiner is viewing the claims, however for mere illustrative purposes the following alternative interpretation is offered as another example of how the claims may be met. Column 7, lines 39-42 show that instructions are stored in (or read into) reservations stations. The section further shows that the instructions are evaluated for dependencies. The section then shows that these instructions are delayed appropriately based on these dependencies, which one of ordinary skill in the art would recognize to mean that instructions are scheduled or selected based on the existing dependencies.

43. Applicant also argues that Nemirovsky does not disclose generating a predicted address based on a predetermined one of the read instructions as given in claim 21. As given in column 6, lines 9-16, Nemirovsky shows that in some embodiments priority determination is the result of branch prediction. Column 1, lines 56-60 further show that branch prediction units are used in the art of the invention. In order to have a branch prediction, a branch instruction must inherently be fetched or read as given above. The included dictionary definition of "branch prediction" shows that it is guessed or predicted whether a branch will be taken or not and code is prefetched from the appropriate location (address). Thus the destination address is predicted (generated) and instructions are executed from that location based on having read and evaluated a branch instruction.

44. Applicant has argued with respect to claim 1 that Nemirovsky does not disclose "a program counter corresponding to a location in each of the instruction memories from

which instructions are read" but in contrast discloses that each stream "has a context frame containing a program counter and register file for that stream". As indicated in the rejection and by Applicant in his arguments, the Examiner has conceded that Nemirovsky does not disclose a plurality of memories and since there is only a single memory (instruction cache memory, for example), Nemirovsky does address each of the instruction memories (there being only one) in his disclosure and it is left to the 103 combination to define the multiple memories and the interaction with them. Further, the fact that Nemirovsky has a single program counter for each stream would not disallow a 103 combination with a reference teaching a program counter corresponding to a location in each of a plurality of instruction memories unless each stream had its own independent instruction memory. Where multiple instruction memories contain instructions for multiple streams, such a combination would be perfectly valid and is accomplished as set forth above in the rejection.

45. Applicant's arguments with respect to claims 10 and 27 have been considered but are moot in view of the new ground(s) of rejection. The Examiner would nonetheless like to address arguments of claim 27.

46. Applicant argues with respect to claim 27 that Nemirovsky does not disclose means for simultaneously reading a plurality of processing instructions from instruction memory that relate to processing of a packet since the instruction streams of Nemirovsky appear to be independent streams. While, it has been found that Nemirovsky does not explicitly disclose simultaneously reading the instructions, this is in no way impacted by the simple fact that Nemirovsky uses independent program

threads. All that needs to be disclosed to fit the Examiner's interpretation is that the system fetches multiple instructions at once. Therefore the above 35 USC 103 rejection that combines Nemirovsky with a teaching of simultaneous fetching of multiple instructions is perfectly valid. Further, column 5, lines 10-21 shows that Nemirovsky's invention is related to packet processing. Applicant has also argued that Nemirovsky does not disclose selecting one of a plurality of read instructions based on a priority encoding of evaluation results related to each of the read instructions. As discussed above, Nemirovsky does disclose selecting one of a plurality of read instructions based on evaluation results related to each of the read instructions. Column 7, lines 39-67 show that the selection of streams (and the instructions therein) is performed based on a priority or priority encoding of these streams.

Conclusion

47. Applicant's amendment necessitated the new ground of rejection for claims 1-4, 6-20, and 23-30 presented in this Office action. The arguments for the other claims were not found to be persuasive. Accordingly, **THIS ACTION IS MADE FINAL**. See MPEP § 706.07(a). Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date of this final action and the advisory action is not mailed until after the end of the THREE-MONTH shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any

extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than SIX MONTHS from the date of this final action.

48. The following is text cited from 37 CFR 1.111(c): In amending in reply to a rejection of claims in an application or patent under reexamination, the applicant or patent owner must clearly point out the patentable novelty which he or she thinks the claims present in view of the state of the art disclosed by the references cited or the objections made. The applicant or patent owner must also show how the amendments avoid such references or objections.

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Shane F Gerstl whose telephone number is (571) 272-4166. The examiner can normally be reached on M-F 6:45-4:15 (First Friday Off).

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Shane F Gerstl
Examiner
Art Unit 2183

SFG
January 11, 2005

EDDIE CHAN
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100